

Automatic Refinement of Hidden Markov Models for Speech Recognition

by

Stephen Schlueter

Submitted to the Department of Electrical Engineering
and Computer Science in Partial Fulfillment of the
Requirements for the Degrees of

Bachelor of Science in Electrical Science and Engineering and
Master of Engineering in Electrical Engineering and Computer Science
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1997

© Massachusetts Institute of Technology, 1997. All Rights Reserved.

Author
Department of Electrical Engineering and Computer Science
May 23, 1997

Certified by
Louis Braid
Professor of Electrical Engineering
Thesis Supervisor

Accepted by
Arthur C. Smith
Chairman, Departmental Committee on Graduate Theses

OCT 29 1997

Automatic Hidden Markov Model Refinement for Speech Recognition

by

Stephen Schlueter

Submitted to the Department of Electrical Engineering and
Computer Science on May 23, 1997, in partial fulfillment of the
requirements for the degrees of Bachelor of Science in Electrical
Science and Engineering and Master of Engineering in Electrical
Engineering and Computer Science

Abstract

Modern speech recognition systems require models well matched to the speech being analyzed. Parameters of speech models vary significantly among individual speakers. Using speaker-independent parameters rather than parameters estimated for the specific speaker generally results in lower recognition accuracy. Manually creating speaker-dependent speech models is tedious and impractical for many applications. This thesis describes a system that uses the orthography of the training data to automatically produce accurate speaker-dependent speech models. Specific problems addressed include generation of multiple pronunciation hypotheses from the utterance and iterative re-estimation of the speech models.

Thesis Supervisor: Louis Braid

Title: Professor of Electrical Engineering

Table of Contents

1	Background	10
1.1	Speaker Independence vs. Speaker Dependence	10
1.2	Goals of This Thesis and Outline	13
2	System-specific Details	16
2.1	Recognizer and Model Training Tools	16
2.2	HMMs and Pronunciation Networks	17
3	Automatic Transcription	20
3.1	Overview of the Automatic Transcription Process	20
3.2	Details of Pronunciation Network Generation	22
3.3	Details of Phonological Rule Generation	25
3.4	Analysis of Resulting Transcriptions	31
4	Experimental Results	38
4.1	Summary of the Data Used	38
4.2	Optimal Network Generation	38
4.3	Iterative Generation of Speaker-dependent Models	43
4.4	Conclusions and Suggestions for Further Study	47
Appendix A Symbol Definitions		50
A.1	HTK Network Symbology	50
A.2	TIMIT Phones	50
Appendix B Confusion Matrices		54
B.1	Replacement Confusion Matrix	56
B.2	Context Confusion Matrix for the Phone “eh”	57
Appendix C Phonological Rules		58
C.1	Rule Format	58
C.2	Manually-chosen Rules	60
Bibliography		62

List of Figures

Figure 1.1: Spectrograms and phonetic transcriptions of two different speakers.....	11
Figure 1.2: Typical steps for speaker-dependent recognition.....	13
Figure 1.3: Automatic speaker-dependent model generation	14
Figure 2.1: Relationship between original models and re-estimated models	17
Figure 2.2: Pronunciation network representation for the word “tomato”	18
Figure 2.3: Alternate network representations for the word “tomato”	18
Figure 3.1: Basic steps of automatic transcription.....	21
Figure 3.2: Simple representations for the phrase, “did you get tomatoes.”	23
Figure 3.3: Representation of a sample network	24
Figure 3.4: A sample network that shows adjacent-phone effects.....	25
Figure 3.5: Optimal alignment of phone sequences.....	26
Figure 3.6: Computation of rule-generating statistics.....	29
Figure 3.7: Steps taken for automatic phonological rule generation	30
Figure 3.8: Automatic transcription using network-assisted recognition	31
Figure 3.9: An example of optimal alignment of two phonetic transcriptions	32
Figure 3.10: Matrix used for matching phone symbols of two different sequences.....	33
Figure 3.11: Waveform with manual and automatic transcriptions.....	37
Figure 4.1: Automatic transcription correctness.....	42
Figure 4.2: Recognition rates vs. number of training iterations	44

List of Tables

Table 4.1: Thresholds for Rule Production.....	43
Table A.1: HTK Network Symbology.....	50
Table A.2: TIMIT phones	51

Chapter 1

Background

Most successful automatic speech recognition algorithms represent speech as a stochastic process. Probabilistic models of individual speech elements are constructed during a training phase, and new sequences of speech elements are compared against the models during a testing or recognition phase. For recognition to be accurate, parameters of the models must be reliably estimated from the training data. To estimate these parameters, each segment of the speech waveform must be assigned to a speech element. This assignment is often done manually by skilled phoneticians. Standard databases for automatic speech recognition have been created in this way (e.g. TIMIT, Resource Management Database, etc.). While these large databases provide an excellent resource for generating models of speech, the effort required to construct them can be daunting.

1.1 Speaker Independence vs. Speaker Dependence

Ideally a speech recognizer comprising a trained set of phone models would convert a novel speech waveform into a phone sequence that matches the sequence of speech sounds contained in the waveform. When this recognition procedure is performed with models derived from a single speaker and the speech waveforms being recognized are produced by that same speaker, recognizers generally produce more accurate results than when phone models based on one speaker are used to recognize speech produced by another speaker. This is due to differences between speech characteristics across speakers

As an example, Figure 1.1 shows spectrograms of the word “tomato” produced by two different speakers. The phonetic transcriptions show how pronunciations can vary across speakers. Even in regions where both speakers produced the identical phone, the two waveforms display different frequency characteristics for different durations. For instance, the top speaker shows a tendency to have much stronger 3rd and 4th formants. This suggests that the phone models for these two speakers would not be the same.

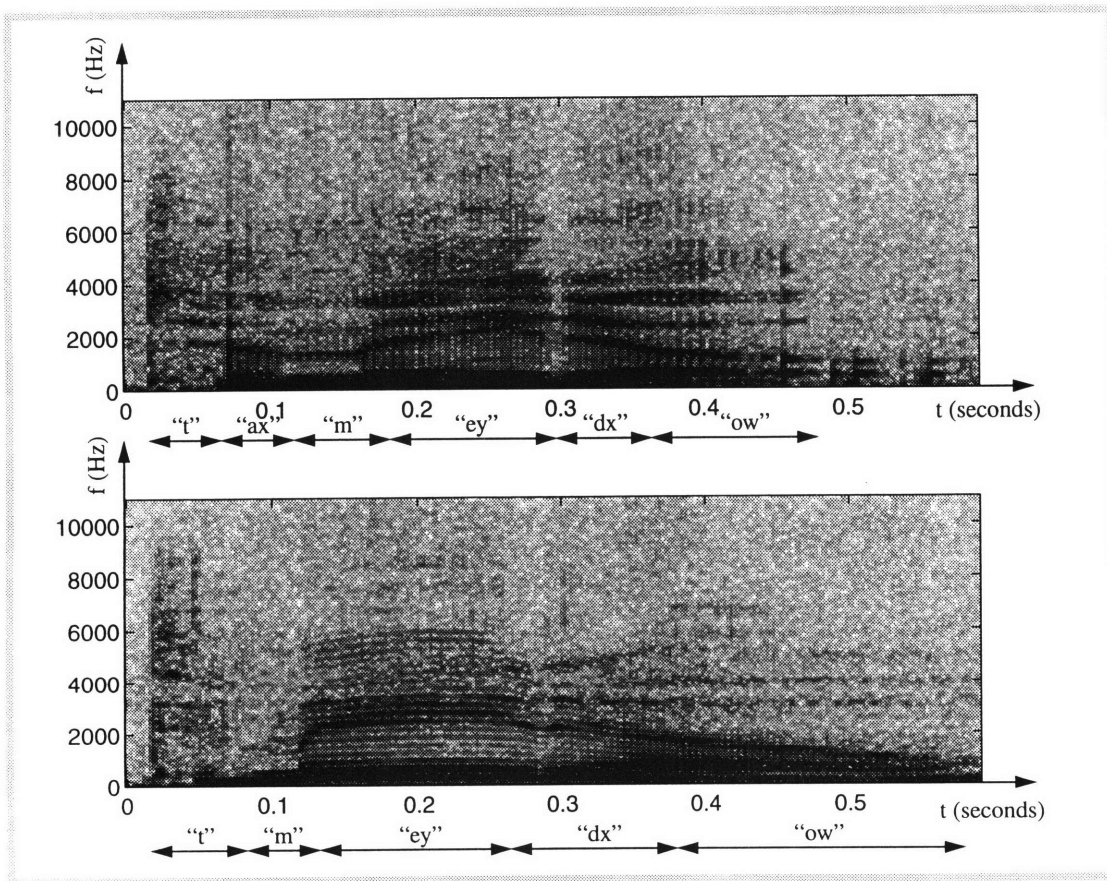


Figure 1.1: Spectrograms and phonetic transcriptions of two different speakers saying the word “tomato.” (See Appendix A.2 for definitions of the phones used in these transcriptions)

In order to incorporate some of the variability among different speakers, models can be estimated from speech produced by a number of speakers. Given a large enough pool of speakers for training, the resulting “speaker-independent” models in general perform similarly for any speaker. Recognizers using speaker-independent models, however, usually do not perform as well as recognizers using “speaker-*dependent*” models, which are models based only on the speaker being recognized. For example, a recognition experiment performed using speaker-independent models based on the training portion of the TIMIT database (3696 sentences from 426 speakers) correctly identified approximately 65% of the phones in a testing portion of the database (480 sentences produced by 160 speakers). In comparison, when recognition was performed with models based on training sentences produced by the individual speaker, roughly 80% of phones within the test sentences were correctly identified.

Figure 1.2 shows the typical steps taken during speaker-dependent recognition. In order to create accurate speaker-dependent models, a large amount of speech data from the individual speaker must be transcribed (divided into segments that represent individual phones and labeled phonetically). Manually labelling the phones within speech waveforms is very time consuming and often impractical. In addition, a large amount of speech data from a single speaker may not be available or is inconvenient to obtain.

Speaker-independent models are more easily obtained since no new training data is required when speech from a new speaker is to be recognized (all training is performed on sentences from a large, existing speech database), but a wealth of empirical data demonstrates that the speaker-independent models are inferior to speaker-dependent models when only a single speaker is to be recognized.

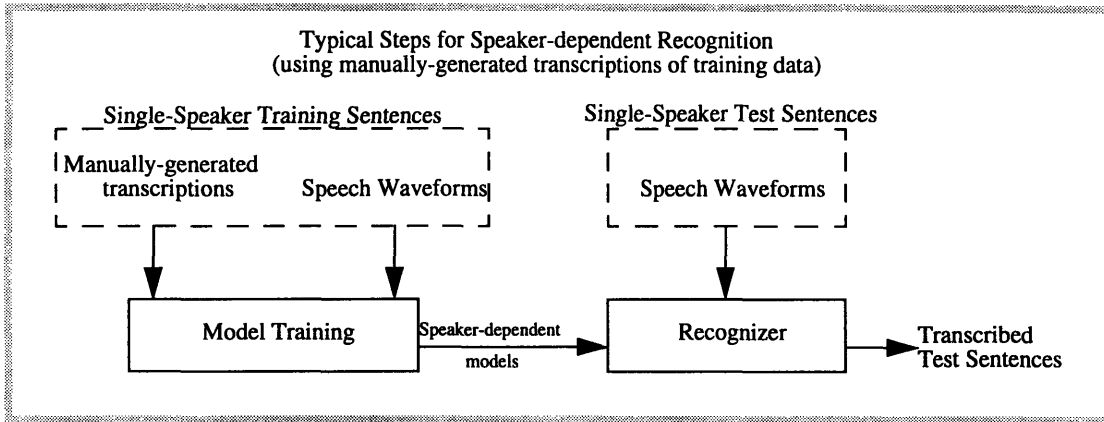


Figure 1.2: Typical steps for speaker-dependent recognition (using manually-generated transcriptions of training data).

1.2 Goals of This Thesis and Outline

This thesis describes a system that automatically creates accurate speaker-dependent models without requiring manual transcriptions of the new speaker's training data. The system instead uses speaker-independent models and the orthography of the training sentences to transcribe the training waveforms automatically. From these transcribed waveforms, speaker-dependent models are estimated.

Steps taken by the system include *pronunciation network generation* (automatic estimation of the possible phone sequences that correspond to a particular orthography), *network-assisted training-sentence transcription* (identifying the segments of a training waveform that correspond to individual phones specified in the pronunciation network), and *speaker-dependent model generation and refinement* (the process by which transcribed training sentences are used to estimate speaker-dependent speech models, and

how these resulting models can be used iteratively to create better speaker-dependent models). Figure 1.3 shows an outline of the relationships between these steps.

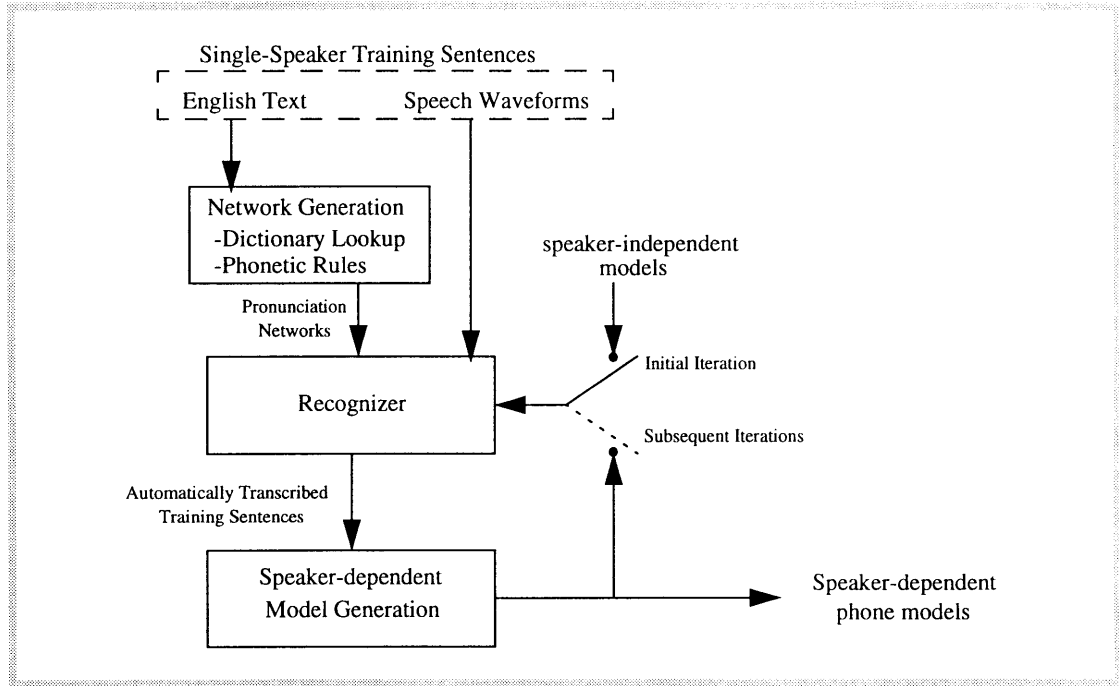


Figure 1.3: Automatic speaker-dependent model generation

The body of the thesis will describe each step in detail. Data produced at intermediate steps will be analyzed to verify the performance of each segment of the system. Finally, the quality of the resulting models will be summarized, and possible ways to improve results of future experiments will be suggested.

Chapter 2

System-specific Details

2.1 Recognizer and Model Training Tools

All experiments described here used a general-purpose hidden Markov model recognition system implemented with routines of Entropic Research Lab's HTK: Hidden Markov Model Toolkit [1]. Rabiner and Juang [2] explains many of the basic concepts of HMMs, as well as Viterbi and Baum-Welch algorithms which were used by the automatic transcription system. The phone models generated by HTK used mel-frequency cepstral coefficients with 12 delta-coefficients to represent each hidden Markov model (HMM) state vector. Each phone model consisted of 3 observation-vector-emitting states. Output probability density functions were represented as a mixture of 6 Gaussians. Though the model-generation procedure described in this thesis is not limited to one particular type of recognition system, the quality of the resulting models may vary depending on the type of recognition system used.

Except for the initialization stage, the hidden Markov models were always trained using the Baum-Welch re-estimation algorithm. The HTK Reference Manual [1] describes Baum-Welch formulae in full detail. This re-estimation method uses transcribed training data to produce the new phone models (See Figure 2.1). The models used for producing the automatic transcriptions are only indirectly related to the new models. Actual model parameters from the original models are discarded during creation of the new models.

Consequently, a fairly large number of training sentences is required in order to produce speaker-dependent models of reasonable quality.

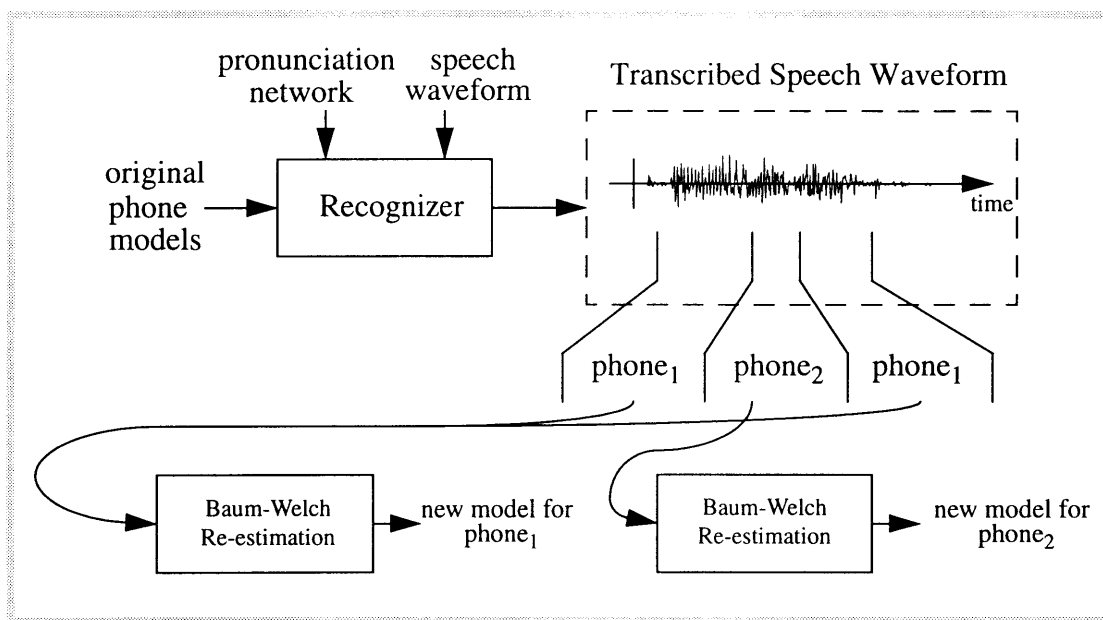


Figure 2.1: Relationship between original models and re-estimated models

2.2 HMMs and Pronunciation Networks

HTK symbology is used to describe models and pronunciation networks. The model for each phone is represented by a TIMIT label (see Appendix A.2 for TIMIT phone definitions). For example, “aa” symbolizes the phone produced by the “o” in the word “hot.” A pronunciation network represents a sequence of models. Each sequence has a starting point and an ending point, but may have multiple paths in between. For instance, the word “tomato” has two common pronunciations, “t ow m ey t ow” and “t ow m aa t ow.” A state-diagram for the transitions between these phones is shown in Figure 2.2,

along with an equivalent HTK representation (Appendix A.1 describes the conventions used by HTK for representing networks).

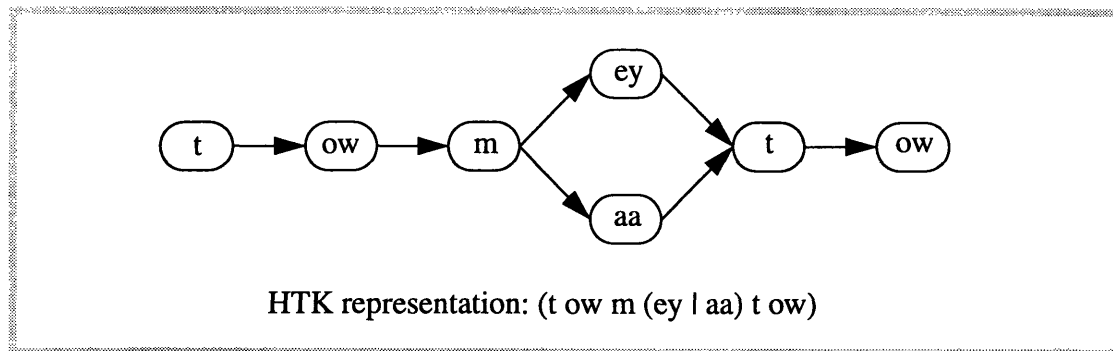


Figure 2.2: Pronunciation network representation for the word “tomato”

For the sake of simplicity, this thesis will often combine sequential members in a state diagram, as shown in Figure 2.3.

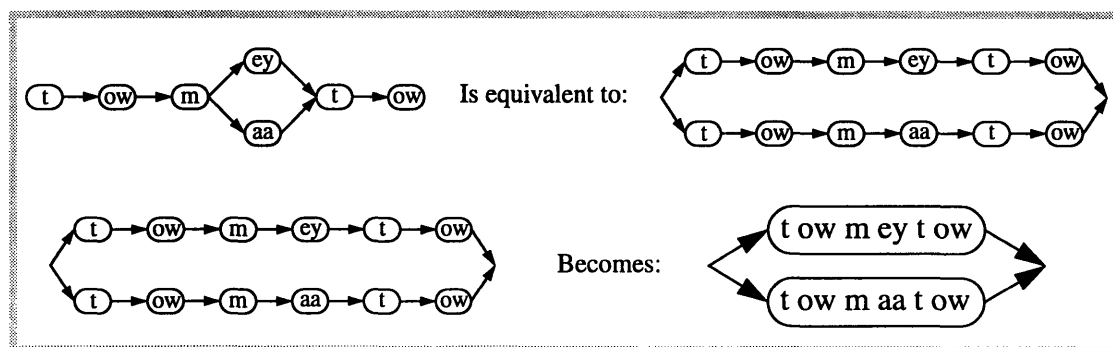


Figure 2.3: Alternate network representations for the word “tomato”

Notice that different network representations can represent the same underlying logical sequence of symbols. The state diagrams of “ $(p_1 (p_2 | p_3) p_4)$ ” and “ $((p_1 p_2 p_4) | (p_1 p_3 p_4))$ ” are not the same, but the resulting paths that can be followed pass through identical sequences of models and are thus functionally equivalent. To minimize confusion, the form most suited for illustrating each particular concept will be used.

Chapter 3

Automatic Transcription

In order to create the best possible models from training sentences, the exact phone sequences present in the speech samples of these sentences must be known. The time-boundaries of each phone must also be given to the model re-estimation system. The knowledge normally used to obtain these pieces of information includes the acoustic waveforms and orthography (text) of the sentences. Manual labeling of the speech waveforms involves listening to segments of audio and deciding which phone each segment of audio represents and where the boundaries between phones exist. This manual process is both time-consuming and subjective, making it impractical for many applications.

3.1 Overview of the Automatic Transcription Process

An automatic algorithm that mimics the manual transcription process can be broken down into two major steps. First the text is translated into a "pronunciation network," a representation of all possible phone sequences that might exist in the speech sample. For example, the word "tomato" might correspond to a phone network that contains two pronunciations, "t ow m ey t ow" and "t ow m aa t ow." Second, a recognizer based on speaker-independent speech models chooses one of the possible phone sequences from the network and estimates the time boundaries of each phone in the sequence. The network forces the recognizer to choose among a small number of phones that are likely to be

present, so the recognizer is able to accurately determine the time boundaries of each phone. Figure 3.1 shows the relationships among the above steps.

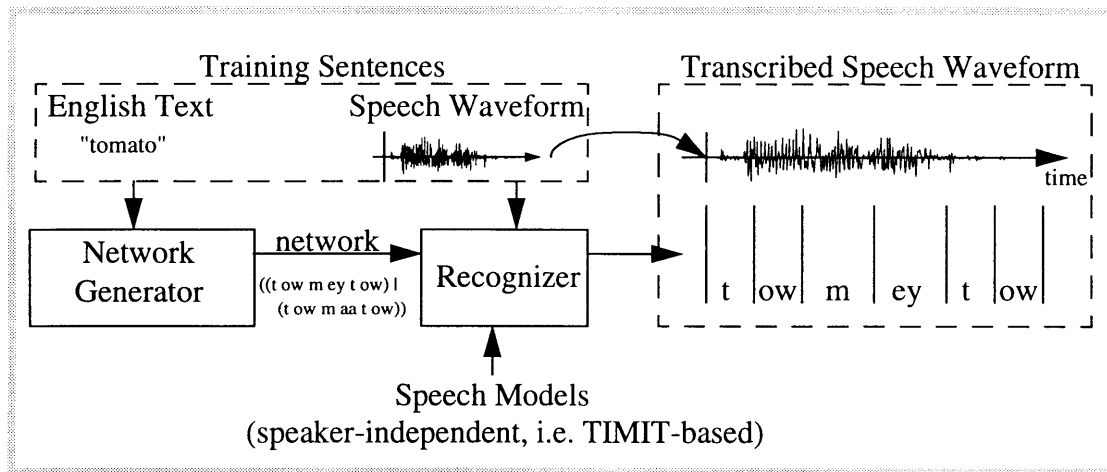


Figure 3.1: Basic steps of automatic transcription

Since models that are tuned to the training sentences do not yet exist (creation of such models is the goal of the entire system), the recognizer must use a set of speaker-independent phone models. The system that was developed used a set of hidden Markov models based on 3696 TIMIT sentences from 426 different speakers. Using speaker-independent models, a recognizer would be expected to produce poor results, but since the pronunciation networks limit the recognizer's choices to likely phones sequences, the results are much more accurate.

Besides choosing among alternate pronunciations specified by the networks, the job of the recognizer is to find *when* the phones occur. If the network is well constrained so that the recognizer does not have a multitude of phones from which to choose, speaker-independent models (such as the TIMIT-based models) should be adequate for time-alignment.

3.2 Details of Pronunciation Network Generation

When told the exact phone sequence that is present within an audio sample, the recognizer can determine reasonable estimates for the time boundaries between phones. The network for an utterance would therefore ideally include only a single phone sequence that identically matches the underlying phones. Generation of such a network is usually not possible since a single phrase of English text can often be pronounced many different ways. Instead, the network generation system that was developed attempts to create a network that contains all likely phone sequences while keeping the network as constrained as possible.

A simple dictionary lookup procedure was used to produce the framework of the networks. The dictionary used was the Carnegie Mellon University (CMU) 116000-word phonetic dictionary [3]. The CMU dictionary includes formal words and plurals, as well as many possessives, abbreviations, proper names, and slang words.

Though the sentences that were used contained only one word that was not in the dictionary (“piney,” for those of you who are curious), an algorithm that estimates the phone sequences of an unknown words was included in the system. This algorithm used a fixed set of phonological rules, applied from left to right, to estimate the phone sequence of each unknown word. This method of phone sequence estimation is not highly accurate, but it was necessary to provide the system with some means of handling words not found in the dictionary. In all experiments performed, the impact of this subsystem was statistically insignificant.

If the translation from text to phones were a simple one-to-one mapping, dictionary lookup would complete the network-generation procedure (except, of course, for words not present in the dictionary). An example of a simple network, generated by direct dictionary lookup, is shown in Figure 3.2.

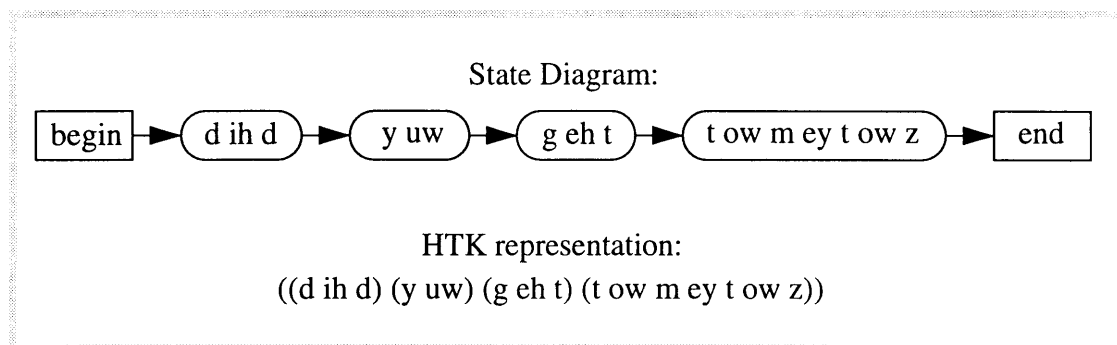


Figure 3.2: Simple representations for the phrase, "did you get tomatoes."

One complication addressed was alternate pronunciation of words. The CMU dictionary provides multiple pronunciations where appropriate. This made the CMU dictionary especially well suited for the task at hand. While the use of each pronunciation does not truly occur with the same probability, networks gave equal weighting to all pronunciations for a given word.

A second complication addressed was spacing between words. Silence may or may not exist between words in an utterance, depending on the person speaking, the words being spoken, and the rate at which the words are spoken. Figure 3.3 shows an example of how alternate pronunciations and inter-word silences were represented by the system.

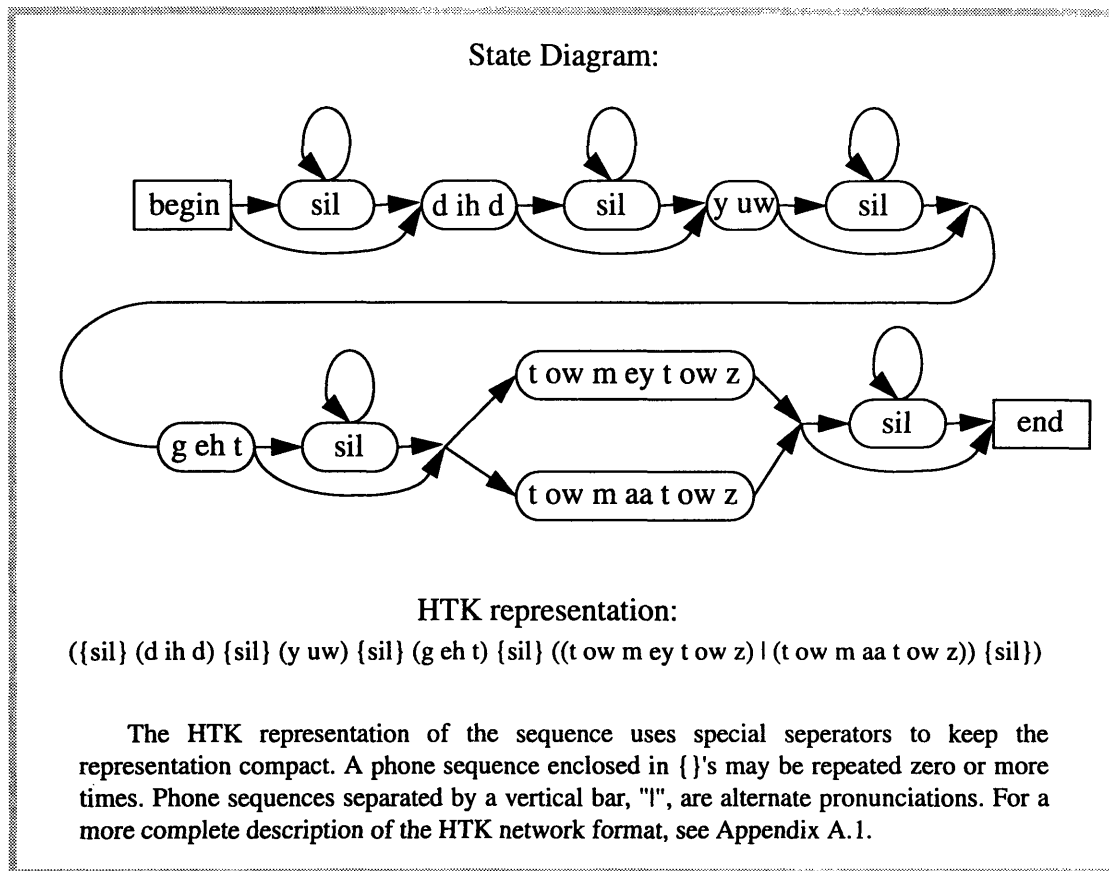


Figure 3.3: Representation of a sample network produced from the phrase, “did you get tomatoes”

The final complication addressed was the consequence of placing words adjacent to each other in continuous speech. For instance, the phrase "did you" is often pronounced as "dijoo" (see Figure 3.4). Another example is the phrase, “get tomatoes,” in which only one of the adjacent “t”s may be heard.

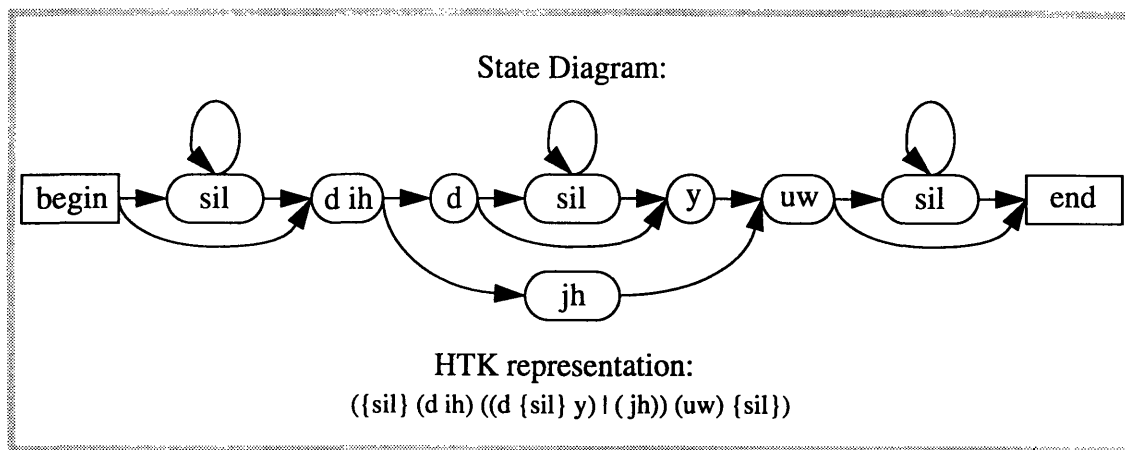


Figure 3.4: A sample network that shows adjacent-phone effects in the phrase, “did you.”

The network generation system used a set of phonological rules to identify contexts in which adjacent phones affect one another. The rule generation system allowed both addition and elimination of options from the pronunciation network. Rather than using a static set of rules chosen by phoneticians (e.g. as described by Cohen and Mercer [4] and Oshika et al [6]), rules were created by a program that compared the performance of the unmodified automatic transcription system with manually generated transcriptions of the TIMIT database. This allowed the resulting rules to compensate for situations in which the unmodified automatic transcription system performed poorly, a benefit that would not result from the use of a static set of rules chosen by phoneticians.

3.3 Details of Phonological Rule Generation

To generate the phonological rules, the automatic transcription procedure (Figure 3.1) was first performed on a group of 450 TIMIT training sentences without applying any rules during network generation. The transcription for each sentence is a single phone

sequence with the time-boundary information for every phone in the sequence. For rules generation, the time-boundary information was ignored. The resulting phone sequence was then aligned as well as possible with a manually-generated transcription for the same sentence. "Alignment" in this case means associating a phone from the automatically generated transcription with a phone from the manually generated transcription in a manner that minimizes the number of mismatches. Section 3.4 describes the alignment procedure in detail. Figure 3.5 shows an example of this type of alignment.

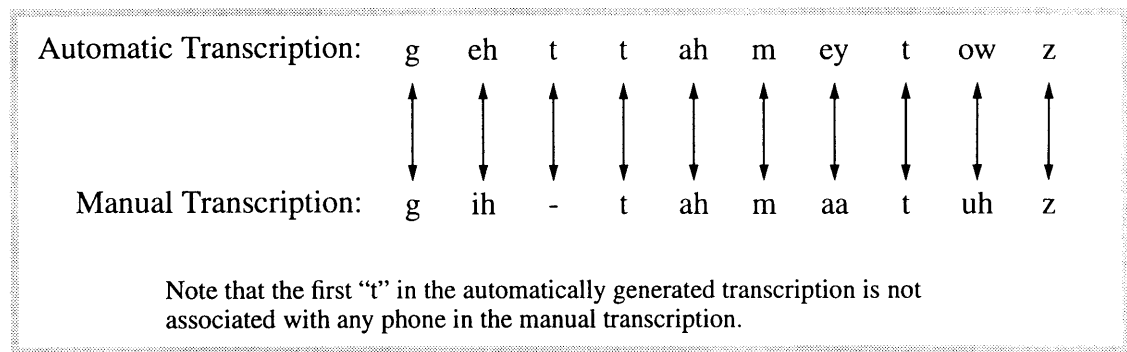


Figure 3.5: Optimal alignment of phone sequences

3.3.1 Producing Rules that Add Network Options

To produce the rules, contexts in which these two transcriptions differed were examined. For example, the automatic transcription in Figure 3.5 contains two adjacent "t" phones. In the manual transcription, only one "t" occurs. This suggests that if there are two adjacent "t"s one of them should become optional.

When determining the context in which a mismatch occurred, only the phone preceding the mismatch and the phone following the mismatch were examined. This

limited range of context sensitivity was suggested by Giachin et al [5]. Limiting the contextual range greatly reduces computation times and rule complexity.

The alignment of “ey” with “aa” in Figure 3.5 is an example of a situation where a limited-context-range rule might be applied. Since the “ey” phone of the automatic transcription is mismatched with the “aa” phone of the manual transcription, the rule-generating system would note the phone preceding the “ey” phone (“m”) and the phone following the “ey” phone (“t”). If the rule generation system finds a sufficient number of instances where “ey” of the automatically-generated phone sequence “m ey t” is aligned with “aa” of the manual transcription, the rule, “allow ‘m aa t’ instead of ‘m ey t’” is created. The exact numerical thresholds that define a “sufficient” number of instances are discussed below.

To compute the statistics required for rule generation, every automatically generated phone, x_n , was examined, along with the preceding and subsequent phones, x_{n-1} and x_{n+1} respectively. These phones were compared to the corresponding manually-generated phones y_{n-1} , y_n , and y_{n+1} .

automatically-generated sequence:	x_{n-1}	x_n	x_{n+1}
aligned manually-generated sequence:	y_{n-1}	y_n	y_{n+1}

Note that the x’s and y’s represent the aligned phones. In situations where phones are inserted or deleted (as the first “t” in Figure 3.5), the x’s and y’s can represent a “null phone” (as signified by “-” in Figure 3.5). Situations where y_n is a null phone correspond

to automatic transcriptions that contain an insertion. This can result in a rule that deletes x_n from the network. Instances where any of the x 's are null phones represent deletions in the automatic transcription. In this case, a wider range of context would need to be examined to extract any useful information from the transcriptions. Instead, these cases are ignored.

The probability that x_n should be replaced by another phone, x_{no} , was computed for all possible x_{no} (all TIMIT phones) as follows:

$N_{correct}(x_{no})$ = number of times $x_n = y_n$ and $x_n = x_{no}$ when " $x_{n-1} x_n x_{n+1}$ " occurs

$N_{incorrect}(x_{no})$ = number of times $x_n \neq y_n$ and $x_n = x_{no}$ when " $x_{n-1} x_n x_{n+1}$ " occurs

(note that $N_{correct}$ and $N_{incorrect}$ are functions of x_{no} , a particular phone being tested)

Probability that x_n is incorrect and should be replaced with x_{no} :

$$P_{incorrect}(x_{no}) = \frac{N_{incorrect}(x_{no})}{N_{incorrect}(x_{no}) + N_{correct}(x_{no})} \quad (3.1)$$

A rule (to allow $x_{n-1} x_{no} x_{n+1}$ where $x_{n-1} x_n x_{n+1}$ is found) is created when $P_{incorrect} > T$ and $(N_{correct} + N_{incorrect}) > J$, where T is an arbitrary threshold, $0.0 < T < 1.0$, and J is an arbitrary integer threshold, $J > 0$. T is the threshold that controls the percentage of discrepancy above which a rule will be suggested. T 's value should probably be no higher than 0.5 for normal rule production, since values of $P_{incorrect}$ greater than 0.5 mean that the automatic transcription is wrong more often than it is correct for a particular context (a rule should probably be created in this case to replace x_n with x_{no}). Decreasing the value of T increases the number of rules produced.

J controls the minimum number of occurrences of a particular sequence ($x_{n-1} x_n x_{n+1}$) that must be seen before a rule will be created. This threshold is used mainly to eliminate inconsequential rules that would otherwise be created when a random discrepancy occurs between the automatic and manual transcriptions. For instance, if within 10000 sentences the phone sequence “ $p_1 p_2 p_3$ ” is produced twice by the automatic transcription system and it is incorrect on one instance during which the manual transcription specifies sequence “ $p_1 p_x p_3$,” $P_{\text{incorrect}}(p_2)$ would then be 0.5. Without the threshold J, this high value of P would cause the creation of a rule that adds the option “ $p_1 p_x p_3$ ” whenever “ $p_1 p_2 p_3$ ” occurs. While this type of rule will probably not affect the accuracy of the resulting transcriptions since the affected phone sequences occur so rarely, the extra rules produced in this manner increase computational overhead. In addition, since these rules are based on statistically insignificant samples, the rules should be avoided. Creation of these rules can be averted by increasing the value of J. The actual values of T and J that were used are discussed in Section 4.2.3. Figure 3.6 shows the steps for computing N_{correct} and $N_{\text{incorrect}}$.

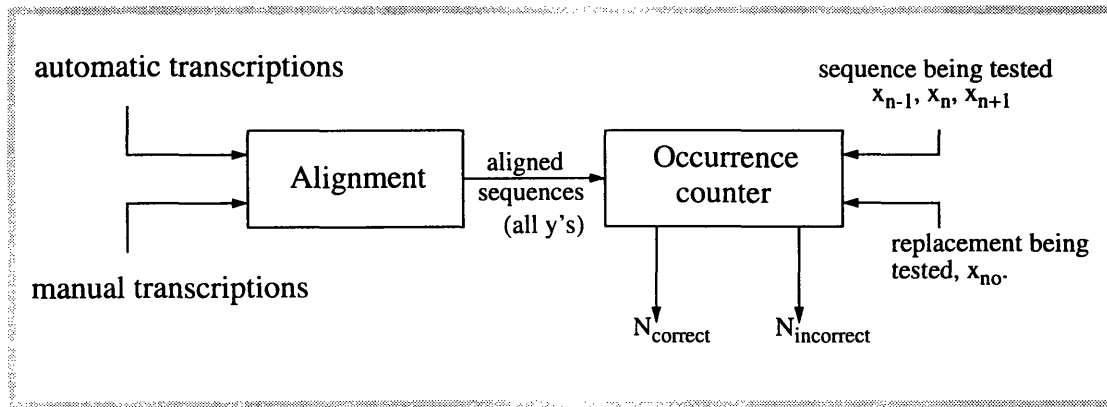


Figure 3.6: Computation of rule-generating statistics

3.3.2 Producing Rules that Remove Network Options

Section 3.3.1 describes how rules that add options to the network are generated. However, situations may arise when the *elimination* of an option is desirable. Perhaps it is found that the automatic network generation procedure always produces a phone sequence "p₁ p₂ p₃" for a particular group of words, while the manual transcriptions almost always specify "p₁ p_a p₃" for the same group of words. In this case, simply adding the option of choosing "p₁ p_a p₃" to the network is not appropriate. Instead, the option of "p₁ p₂ p₃" should be removed and replaced with "p₁ p_a p₃" since "p₁ p₂ p₃" is never (or at least very rarely) present in the manual transcriptions, which are assumed to be perfectly correct. These situations where "option elimination" rules are appropriate can be detected by using high values of the T threshold. Figure 3.7 shows the rule generation process.

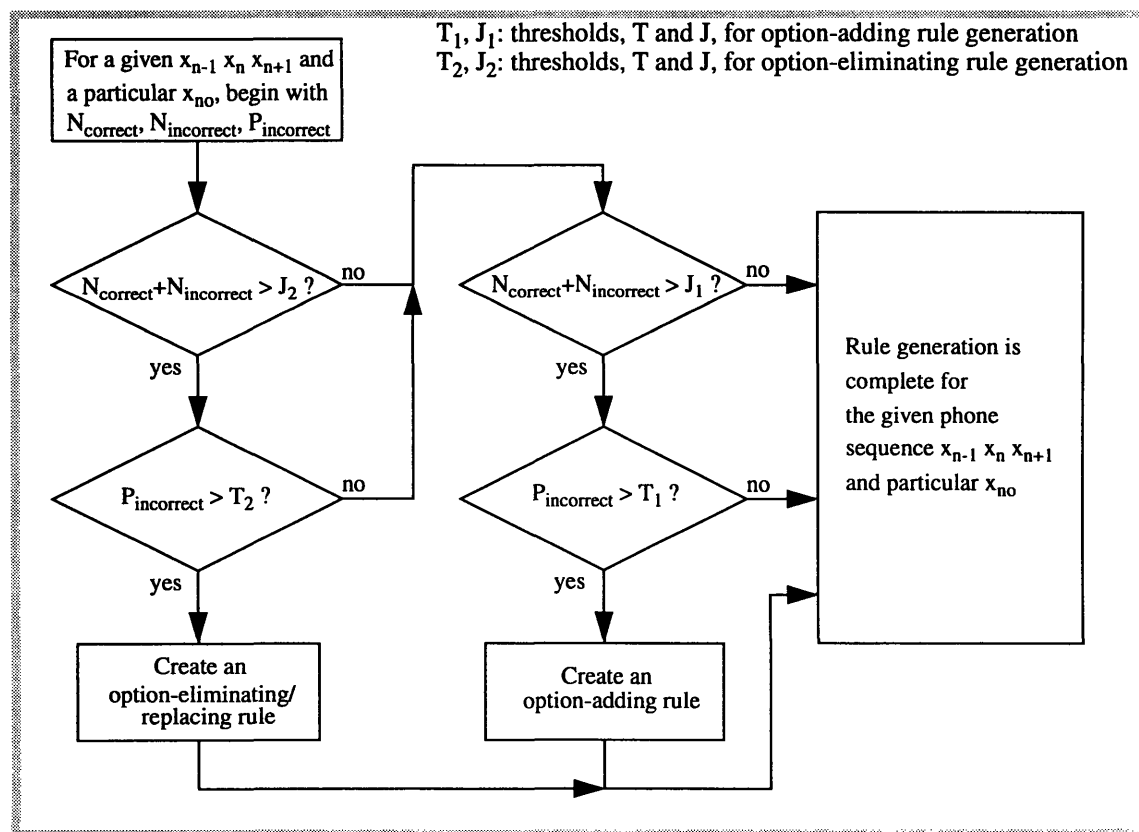


Figure 3.7: Steps taken for automatic phonological rule generation

The entire automatic transcription process is laid out in Figure 3.8, below.

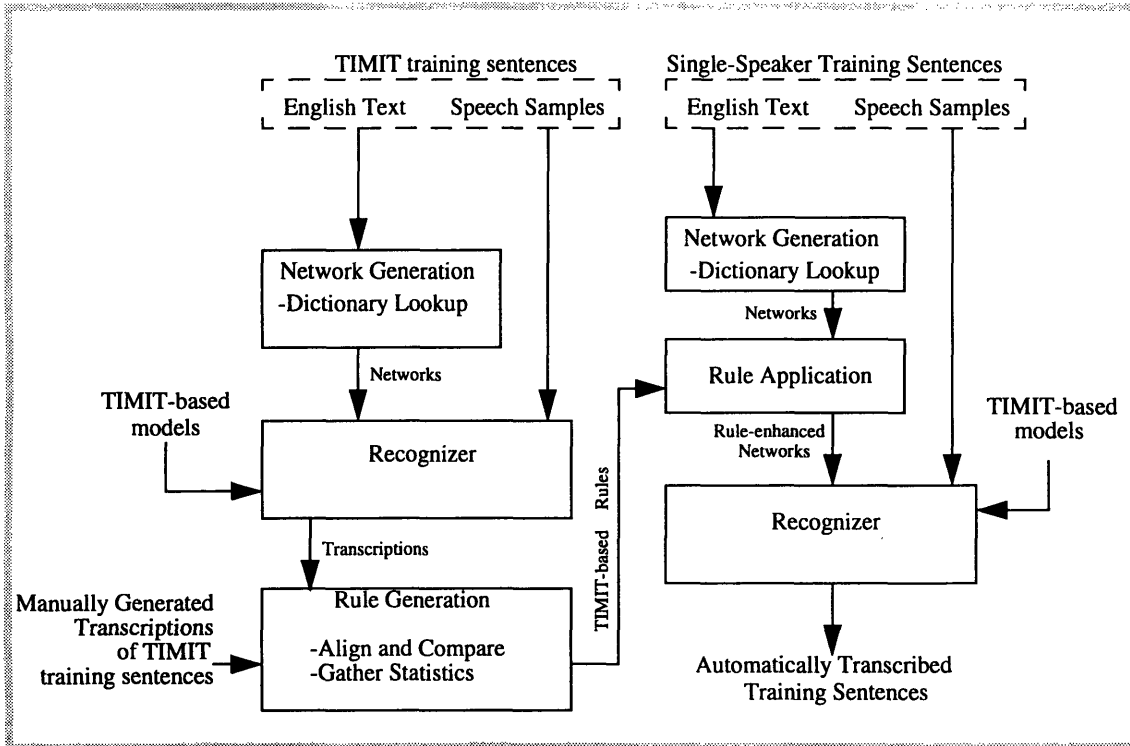


Figure 3.8: Automatic transcription using network-assisted recognition

3.4 Analysis of Resulting Transcriptions

To determine the effectiveness of the automatic transcription process, the transcriptions produced were compared to manually-created transcriptions. This section describes how the comparisons were performed and defines quantitative measures of transcription quality.

Comparison of transcriptions occurs in two places within the system. The rule generation procedure uses transcriptions for sentences from a large database (such as the TIMIT database). Analysis of these transcriptions is a necessary part of the system, as the results of analysis are required to produce the phonological rules described in Section 3.3. Transcriptions are also analyzed at the output of the transcription procedure. Manually-created transcriptions of the output would not be available to a transcription system during real-time operation, but during development of the transcription system the comparison of the system's output to "correct" (manually-created) transcriptions is a useful measure of the automatic transcriber's performance.

3.4.1 Producing an Optimal Comparison

The comparison of the automatically-generated transcription and the manually-generated transcription begins by finding an optimal match between the two phone sequences (See Figure 3.9).

Automatic Transcription, A[i]:	t	ow	m	ey	t	ow	z
Manual Transcription, M[i]:	t	ah	m	aa	t	z	
Optimal alignment ("-" represents absence of a phone):							
A'[i]:	t	ow	m	ey	t	ow	z
M'[i]:	t	ah	m	aa	t	-	z

Figure 3.9: An example of optimal alignment of two phonetic transcriptions

While such an alignment may be easily produced by visual inspection, an automated alignment process must be able to identify quantitatively the best possible alignment. To

accomplish this quantitative analysis, a form of the Viterbi algorithm was used. HTK provides a program (“HResults”) to perform the algorithm, but because data produced during intermediate steps of the algorithm were needed for phonological rule generation, the algorithm was re-implemented. The implementation of the algorithm mimics HTK’s “HResults” program and is described in the following paragraphs.

First, the two transcriptions are arranged as axes of a matrix as in Figure 3.10

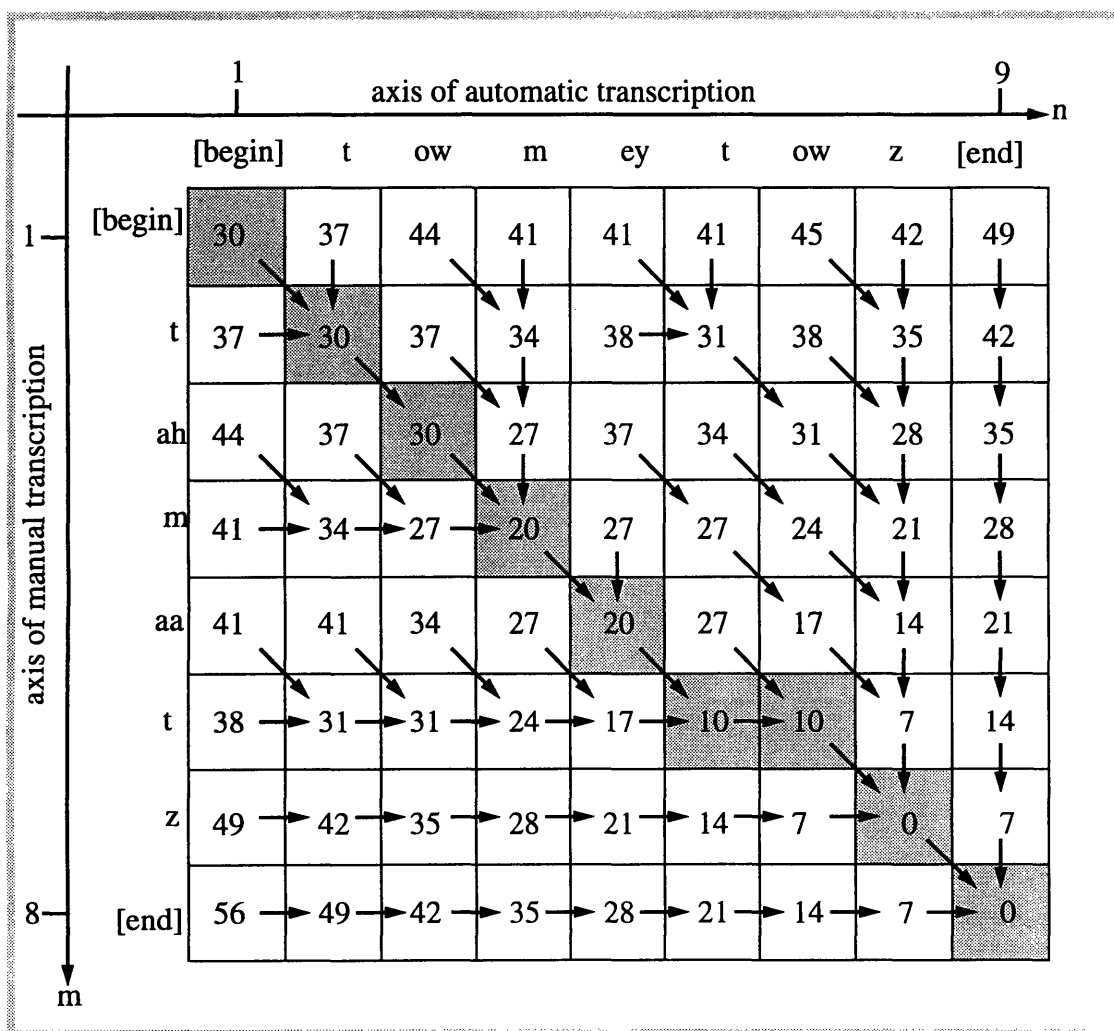


Figure 3.10: Matrix used for matching phone symbols of two different sequences.
(See Section 3.4.1 for explanation)

The value assigned at each coordinate of the matrix represents the “total penalty” for matching the phone along the manual-transcription axis with the phone along the automatic-transcription axis. These penalties are computed in the following manner:

1. Penalty computation begins from the lower-right coordinate of the matrix ($m=8$, $n=9$, or $[8, 9]$ in Figure 3.10).
2. This $[\text{end}, \text{end}]$ coordinate is assigned a penalty of zero.
3. Penalty computation is then performed at all coordinates $[m, n]$ for which penalty computation is complete at $[m+1, n]$, $[m+1, n+1]$, and $[m, n+1]$. Penalties for coordinates exceeding matrix bounds are defined to be infinite (and therefore their computation is always complete). In Figure 3.10, the coordinates that can be computed first are $[8, 8]$ and $[7, 9]$. The penalty value at $[m, n]$ is assigned the sum of the minimum value of $[m+1, n]$, $[m+1, n+1]$ and $[m, n+1]$ plus a value, P , which represents an incremental penalty for that coordinate.

P is dependent upon the transcriptions at $[m, n]$, the “alignment vector” (represented in Figure 3.10 by arrows), and the scoring scheme being used. If the transcriptions match at coordinate $[m, n]$ ($M[m]$ equals $A[n]$), then P is assigned a value of zero (phones match--no penalty). Otherwise P is dependent upon the alignment vector. Each alignment vector points to $\text{argmin}\{[m+1, n]; [m+1, n+1]; [m, n+1]\}$. If the vector is horizontal, it represents an “insertion,” or the automatic transcription contains an extra phone. If the alignment vector is vertical, it represents a deletion-- the automatic transcription lacks a phone that is present in the manual transcription. A diagonal vector represents a substitution, where the automatic and manual transcriptions each contain a phone, but the phones are not the same. HTK software by default uses a symmetric penalty scheme of 7 for insertions, 7 for

deletions and 10 for substitutions. Another common scoring scheme (used by US NIST scoring software) uses values of 3, 3, and 4. All scoring performed by the developed system used the HTK default scoring scheme.

Equations 3.2 and 3.3 summarize the penalty assignment of a single coordinate $[m, n]$ for which penalty computation is complete at $[m+1, n]$, $[m, n+1]$, and $[m+1, n+1]$.

$$P = \begin{cases} 0, (A[n] = M[m]) \\ 7, (A[n] \neq M[m]), \text{insertion or deletion} \\ 10, (A[n] \neq M[m]), \text{substitution} \end{cases} \quad (3.2)$$

$$[m, n] = \text{argmin}\{[m+1, n]; [m+1, n+1]; [m, n+1]\} + P \quad (3.3)$$

4. Finally, step 3 is iterated until penalty values have been computed for every coordinate of the matrix.

By tracing the alignment vectors from the upper left coordinate, $[1, 1]$, of the completed matrix as indicated by the shaded cells in Figure 3.10, an optimal alignment in the form of two new equal-length phone sequences, $M'[i]$ and $A'[i]$, is produced as follows:

Index “i” is first set to zero. Where a horizontal alignment vector enters coordinate $[m, n]$, $M'[i]$ is assigned an “absent phone” value and $A'[i]$ is assigned the value of $A[n]$. Where a vertical alignment vector enters coordinate $[m, n]$, $M'[i]$ is assigned the value of $M[m]$ and $A'[i]$ is assigned an “absent phone” value. A diagonal alignment vector entering coordinate $[m, n]$ signals that $M[m]$ is assigned to $M'[i]$ and $A[n]$ is assigned to $A'[i]$.

The index, “i,” is incremented after traversing each alignment vector. When [end, end] of the matrix is finally reached, $A'[i]$ and $M'[i]$ contain the “optimal” alignment of the automatic and manual transcriptions.

3.4.2 Quantitative Scoring of Alignment

Once the optimal alignment of two phone sequences has been determined, a measure that quantifies how well the two sequences match must be calculated. To accomplish this, we define the following quantities:

H, “hits”: The number of labels in the automatic transcription that match the aligned labels in the manual transcription.

D, “deletions”: The number of labels in the manual transcription that do not correspond to any label in the automatic transcription.

I, “insertions”: The number of labels in the automatic transcription that do not correspond to any label in the manual transcription.

S, “substitutions”: The number of labels in the automatic transcription that are matched with a different label in the manual transcription.

N, “number of phones”: The total number of phones in the manual transcription, equal to $H+D+S$.

From the above quantities, two commonly used metrics, “percent correct” and “accuracy,” can be derived. These two values represent the quality of the automatic

transcription. Equations 3.4 and 3.5, below, show the definitions for “percent correct” and “accuracy.”

$$\%Correct = \frac{H}{N} 100\% \quad (3.4)$$

$$Accuracy = \frac{(H - I)}{N} 100\% \quad (3.5)$$

“Percent correct” is an “optimistic” rating that ignores insertions, while “accuracy” accounts for all possible errors. These different scores exist because some applications are able to cope with insertions easily, while other systems are sensitive to errors introduced by insertions. For completeness, both values will be listed in all results.

Figure 3.11 illustrates how insertion of a phone in a transcription changes the boundaries of neighboring phones.

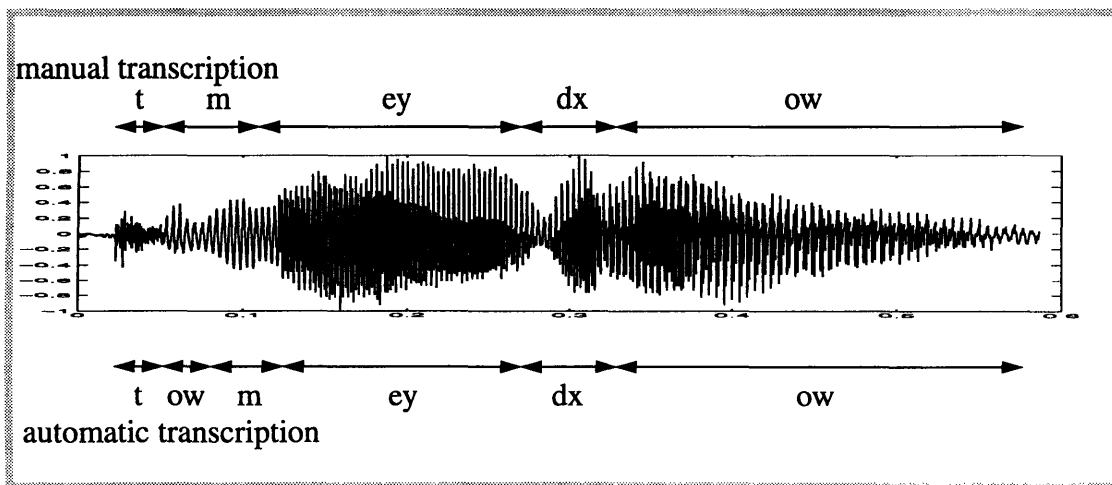


Figure 3.11: Waveform with manual and automatic transcriptions (automatic transcription contains an inserted “ow”).

Chapter 4

Experimental Results

4.1 Summary of the Data Used

In order to test the automatic transcription system, two speech databases were used. The TIMIT database (3696 sentences from 426 different speakers) was used to create speaker-independent phone models and to generate phonetic rules (Section 3.3). The second database (Susan) included speech recorded from a single female speaker. The Susan sentences were divided into two groups of 560 training sentences (roughly 17000 individual phones) and 180 test sentences (roughly 5200 phones). Speech samples, English orthography, and manually-generated transcriptions were available for both databases.

4.2 Optimal Network Generation

4.2.1 Establishing Results for Comparison

Before pronunciation network generation was performed, a recognizer equipped with TIMIT-based models was applied to the Susan training sentences and the resulting transcriptions were compared to manually-generated transcriptions of the sentences. This established the accuracy that can be expected when speaker-independent models are used during unassisted recognition (recognition performed without network data). The

recognition rates that resulted were quite poor¹, 46% correct, and 30% accuracy, as expected.

4.2.2 Improving Recognition Results with Basic Network Generation

Once this base-line performance was established, tests began to determine how the use of pronunciation networks based on sentence orthography would effect recognition rates. Initial networks were generated using only dictionary-lookup of words and optional silence between words. Tests showed that this increased performance to 80% correct and 71% accuracy. While this is a significant increase from unassisted recognition, the remaining error is still quite large.

4.2.3 Determining Phonetic Rules

Before the automatic rule generation system was developed, statistics of the network-assisted transcriptions were inspected manually to determine the source of error. First, a “replacement confusion matrix” was examined. An example of a replacement confusion matrix is shown in Appendix B.1. This type of matrix, with phones from the manual transcriptions along the vertical axis and phones from the automatic transcriptions along the horizontal axis, shows the number of times each particular phone from the automatic transcriptions is matched with phones from the manual transcription. Appendix B explains confusion matrices in more detail.

1. Some of the error may be attributable to the different recording conditions and labeling styles that were used for the TIMIT and Susan databases.

The replacement matrix includes data from all sentences transcribed, but it contains no information about the context in which matches were made. The concentration of high-valued data points along the diagonal axis shows that most of the time the automatic and manual transcriptions chose the same phone.

The replacement confusion matrix was used to determine which phones were producing the most errors. Once this was known, context confusion matrices were produced for each of the error-prone phones. A context confusion matrix (see Appendix B.2) resembles a replacement confusion matrix, except that phones along the vertical axis represent the phone that preceded the error-prone phone and phones along the horizontal axis represent phones that followed the error-prone phone.

Examination of the replacement matrix revealed that most errors were related to a few particular phones. Examination of the context matrices showed that not only were certain phones subject to error, but they were subject to error in particular contexts. This was the basis for generation of phonetic rules (described in Section 3.3).

Phonetic rules were initially created manually in order to determine the effectiveness of rule use. The rules were chosen in an iterative fashion. By observing the confusion matrices, the most obvious rules were chosen (when repeated consonants are found in a network, one of them should become optional, etc.). These rules were chosen somewhat subjectively and were meant to be used only to verify that context rules would improve transcriptions.

By using these rules for network generation and re-transcribing the same sentences, new confusion matrices were formed. The new matrices revealed less obvious errors. This process was repeated through a third iteration. In addition to rules based on the statistics gathered from the confusion matrices, rules suggested by Cohen and Mercer [4] and Giachin et al [5] were also tried. Appendix C shows the final set of 47 rules that were manually chosen.

Transcription was again performed using TIMIT models with the Susan training sentences. During this computation, however, the manually-chosen context rules were applied to the networks. The results were promising, yielding scores of 85% correct and 79% accuracy. With this increase in recognition rates, the quality of phone models based on transcribed data would also increase.

Since the phonetic rules depend only upon the language being spoken, they do not need to be recomputed for each new speaker (the rules chosen were entirely based on TIMIT data anyway). While the performance of the manually-chosen rules was good, a more convenient method of determining rules was needed, perhaps for use with other languages or alternate sets of phones. Automatic rule generation could also help determine whether the manually chosen rules were, in fact, the best rules. This was the motivation for developing an automatic rule-generation system.

The rule generation system described in Section 3.3 was used to generate rules based on 450 TIMIT training sentences. Generation began with high thresholds (T and J, in section 3.3) so that only the most evident errors in automatic transcription would produce new rules. Values of these thresholds are shown in Table 4.1. These values were

established by trial and error, varying T until the number of rules produced was on the order of the number of rules chosen manually. The value of J only prevents inconsequential rules from being created. Its value is not critical for performance, but low values of J cause excess rule generation and computation.

The TIMIT sentences were re-transcribed using the new rules. A second set of rules was then produced using lower thresholds. As can be seen in Figure 4.1, benefits from further iterations of this procedure diminish rapidly after the first iteration. Values of T and

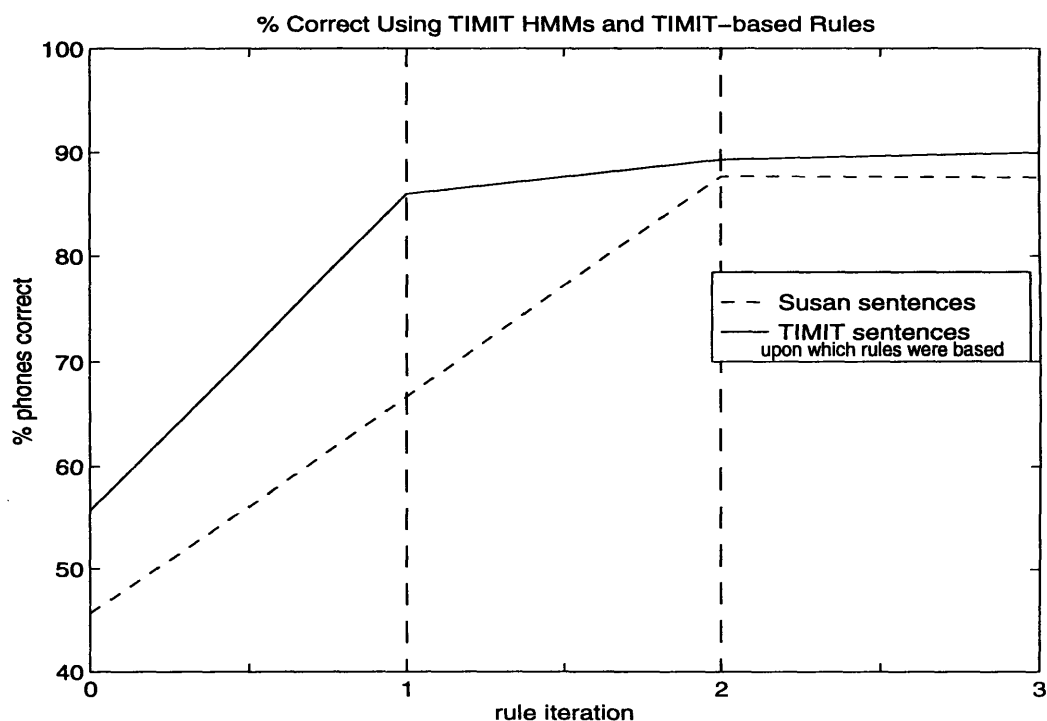


Figure 4.1: Automatic transcription correctness
vs. number of rule-generating iterations

J that were used during each iteration are shown in Table 4.1. The sets of rules produced by

the first two iterations were combined into a single set of rules, hereafter referred to as “TIMIT-based rules.”

Iteration	T_1 (for option-adding rules)	J_1 for option-adding rules	T_2 (for option-eliminating rules)	J_2 (for option-eliminating rules)
0	0.60	5	0.95	5
1+	0.40	5	0.95	5

Table 4.1: Thresholds for Rule Production
(T’s and J’s correspond to those in Figure 3.7)

Automatic transcription performed with the automatically-generated rules resulted in 88% correct and 83% accuracy when applied to the Susan training sentences, a slight increase in performance relative to transcription using the manually-generated rules.

4.3 Iterative Generation of Speaker-dependent Models

When new methods for further improving the network generation system could no longer be found, speaker-dependent model generation was performed. Figure 1.3 gives a general block-diagram of the procedure that was followed.

Using TIMIT-based models, recognition performed on Susan test sentences without the aid of networks produced results of 46% correct and 31% accuracy. Models that were produced after the first iteration of model refinement yielded greatly improved results--

73% correct and 67% accuracy. Unfortunately, as shown in Figure 4.2, results of further iterations were disappointing. Performance actually declined after the first iteration.

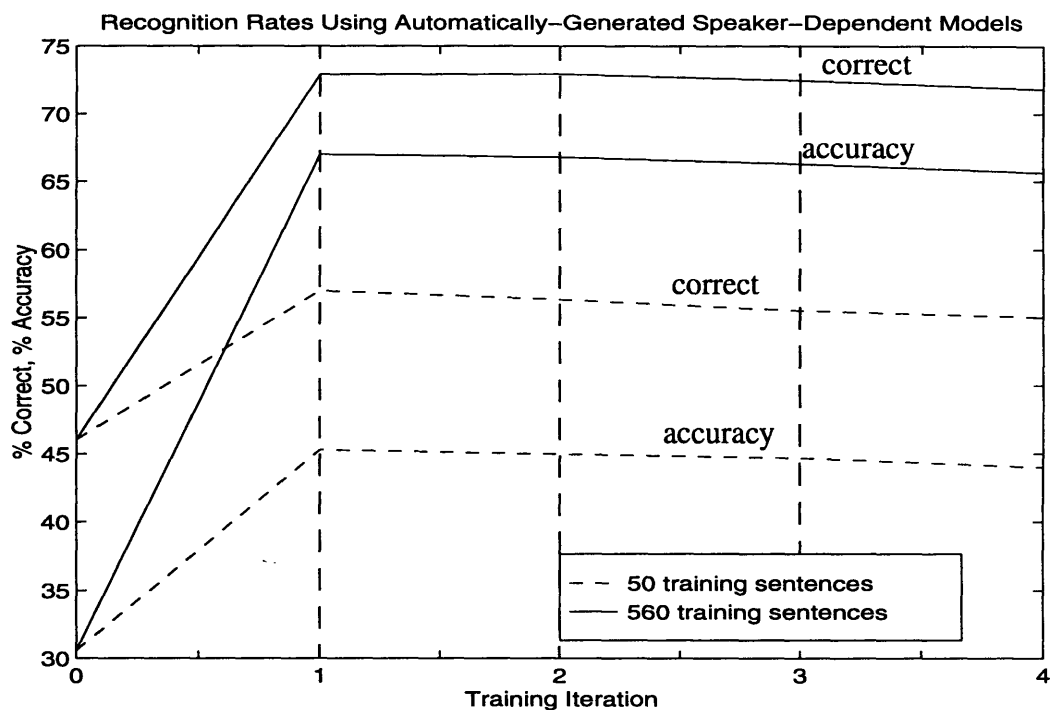


Figure 4.2: Recognition rates vs. number of training iterations
(unassisted recognition-- no networks used)

Figure 4.2 shows recognition rates when compared to manual transcriptions. The same recognition was compared to automatically-generated transcriptions to see whether recognition was improving relative to the automatically-generated transcriptions, but the same plateaus with approximately the same magnitudes as in Figure 4.2 were reproduced. A slight positive slope, however, was observed on the plateaus (between iterations 1 and 4), rather than the slight negative slope seen in Figure 4.2. The total difference in percent correct at iteration 4 relative to Figure 4.2 was less than 1%.

Several possible reasons for the lack of improvement beyond the first iteration were postulated. First, pronunciation networks may be too highly constrained. If the networks do not provide sufficient alternate pronunciations for each phrase, they will limit the quality of training sentence transcriptions. No matter how good the models used for transcription are, there would always be error introduced by the constrained networks. This could account for some of the 12% error during training sentence transcription (the peak performance as shown in Figure 4.1). If the constrained networks are the major cause of the 12% error, then the quality of transcriptions cannot be improved even with better models. In this case, even though the first iteration yielded much improved models, the second iteration would yield approximately the same models. Because the models do not change significantly from one iteration to the next, neither would recognition rates. This would explain the plateaus in Figure 4.2.

Less constrained networks were produced by lowering threshold T_1 to 0.05 for rule generation. Though inspection of the resulting networks verified that the networks contained many more alternative phone sequences, there was no significant effect on the results. Relaxing constraints on the networks after each iteration was also attempted. In every case, the same plateaus of recognition rates as in Figure 4.2 persisted, suggesting (but not ruling out) that constrained networks were not the cause.

The possibility that the models at this point were nearly optimal for the given constraints on model definitions (dimensionality of state vectors, number of states, etc.) was examined, but slightly better models were produced from the same training data using

manual transcriptions. Models based on manual transcriptions of the same Susan training sentences yielded 72% accuracy¹, 5% better than the automatically generated models.

The remaining cause of error is not easily solved. Either the networks generated are too constrained because the context examined by the rules is too narrow (phones may affect not only adjacent phones, but phones more distant in the phrase), or the errors during transcription are not predicable from orthography alone.

4.3.1 The Role of Context-dependent Phone Models

Context-dependent phone models take into account the phones surrounding the phone being modeled. For example, the “ae” phones in the phone sequences “hh ae t,” “hh ae d,” “b ae t,” and “b ae d” are each represented by different models because each falls in a different context. Performance of systems using context-dependent models has been shown to be significantly better than that of systems based on context-independent models. The trade-off, of course, is computational complexity. Recognition performed on n phones with context-*independent* models requires n phone models, while recognition performed on n phones with context-*dependent* models requires n^3 models. To reduce the number of models, right-context-dependent models can be used, which only take into account the phone following the phone being modeled. This results in n^2 models where n phones are used.

1. Models based on manually-generated transcriptions were estimated from 960 sentences, rather than the subset of 560 used here. This would account for some of the 5% gap in performance.

On the Susan database, right-context-dependent phone models were shown to improve recognition rates to about 80% accuracy vs. 72% accuracy for context-independent models. Automatic network-assisted transcription of the Susan training sentences was performed using right-context-dependent phone models, but transcription quality improved by only about 1% in relation to network-assisted transcription using context-independent models. This suggests that use of networks eliminates the same types of errors during recognition that are eliminated by the use of context-dependent models. Because the benefits of using context-dependent models in this case did not outweigh the computational cost, further work using context-dependent models was not attempted.

4.4 Conclusions and Suggestions for Further Study

While the automatic generation of speaker-dependent models was successful, the quality of the resulting models did not match the quality of models based on manually transcribed data. This difference between 67% recognition accuracy for automatically generated models and 72% accuracy for manually generated models is significant and needs further study.

Different phoneticians labeled the Susan and TIMIT databases. Their transcribing styles were likely not identical. Therefore, the rule-generating subsystem may have inferred some pronunciation rules appropriate for TIMIT but not for the Susan database. This could account for some of the superiority of the models generated from hand-labeled data.

As previously mentioned in section 4.3, the manually-generated models that resulted in 72% accuracy during recognition were based on 960 sentences, while the automatically-generated models were estimated from only 560 sentences. This could also account for some of the recognition error.

The remaining error may be attributable to pronunciations not predictable from the orthography of the sentences. Since in these cases the network will not contain the pronunciation present in the waveform, the quality of the resulting models will suffer.

The iterative process of refining the models did not perform as expected. Rather than asymptotically approaching a limit, recognition rates plateaued after the first iteration of refinement. Further experimentation should address this effect. Increasing the number of neighboring phones taken into account during rule generation might improve automatic transcriptions, but the number of rules, and therefore computational complexity, would increase dramatically. A better approach may be to analyze groups of related phone sequences when producing rules, rather than examining phones individually. For example, the phone sequences “ $p_a p_2 p_3$ ” and “ $p_b p_2 p_3$ ” may not occur a sufficient number of times to surpass thresholds T and J , but if p_a and p_b are related (perhaps in both cases the manual transcription uses p_c rather than p_a or p_b , or perhaps both p_a and p_b are vowels) the rule generation system could combine analysis of the sequences to produce a beneficial rule.

Future research should also seek to improve the method by which new models are generated from the automatically-generated transcriptions. By discarding the original models and generating entirely new models based on the transcriptions, the system described in this thesis made necessary the use of a large number of training sentences (as

Figure 4.2 shows, decreasing the number of training sentences greatly reduced performance). Rather than creating entirely new models, a future system could use transcribed data to adjust the original models. This method of model generation might allow the system to attain good results with fewer training sentences.

Appendix A

Symbol Definitions

A.1 HTK Network Symbolology

Table A.1: HTK Network Symbolology

Network Symbol	Meaning
()	symbols enclosed in parentheses are required and must occur sequentially. For example $(p_1 p_2 p_3)$ means that p_1 must occur, followed by p_2 , and finally by p_3 .
{ }	Braces indicate that the enclosed sequence of symbols may occur zero or more times. $(p_1 \{p_2\} p_3)$ indicates that the sequence must begin with p_1 , followed zero or more occurrences of p_2 , and ending with p_3 .
< >	Angle-brackets are the same as braces except the enclosed sequence must occur at least once.
	A vertical bar signifies an “OR” operation. $(p_1 (p_2 p_3) p_4)$ states that the sequence must begin with p_1 , followed by p_2 or p_3 , and terminated by p_4 .

A.2 TIMIT Phones

TIMIT phone symbols are used throughout this text. Since some sources of data used different phone representations, an effort was made to ensure that representations were well-matched and consistent. The most significant discrepancy occurs in the CMU database [3]. Whereas TIMIT defines the phones “ax” and “ah,” the CMU pronouncing dictionary uses only “ah.” The CMU dictionary also lacks any representation for “flapped” d’s and t’s (TIMIT uses the “dx” symbol). The use of context-sensitive phonetic rules

(described in section 3.3) allows the automatic transcription system to “guess” whether “ax” and “dx” phones should be substituted in the CMU database, but some errors may be introduced by this process.

Table A.1: TIMIT phones

phone symbol	definition/example	phone symbol	definition/example
aa	<u>h</u> ot	iy	<u>f</u> ee <u>t</u>
ae	<u>a</u> dd	jh	a <u>g</u> e
ah	b <u>u</u> tt	k	s <u>i</u> ck
ao	<u>o</u> r	l	w <u>i</u> ll
aw	<u>l</u> oud	m	j <u>a</u> m
ax	th <u>e</u>	n	o <u>n</u>
ay	n <u>i</u> ce, <u>i</u> ce	ng	l <u>o</u> ng
b	<u>b</u> uy	ow	g <u>o</u>
ch	<u>ch</u> oose	oy	t <u>o</u> y
cl	<unvoiced closure>	p	<u>p</u> ie
d	<u>a</u> dd	r	r <u>e</u> nt
dh	<u>th</u> is	s	s <u>i</u> ze
dx	b <u>u</u> tt <u>er</u> (flapped to sound like “d”)	sh	w <u>i</u> sh
eh	<u>l</u> et	sil	<silence>
el	bott <u>l</u> e	t	<u>t</u> alk
en	butt <u>o</u> n	th	<u>th</u> in
epi	<epithetical silence>	uh	<u>l</u> ook
er	<u>f</u> ur	uw	b <u>o</u> ot
ey	s <u>a</u> y	v	<u>v</u> ery
f	<u>f</u> ill	vcl	<voiced closure>

phone symbol	definition/example	phone symbol	definition/example
g	<u>g</u> o	w	<u>w</u> alk
hh	<u>h</u> ill	y	<u>y</u> es
ih	l <u>i</u> t	z	<u>z</u> one
ix	ros <u>e</u> s	zh	u <u>s</u> ual

Appendix B

Confusion Matrices

Confusion matrices describe the types of mismatches that occur between two sets of transcriptions. Here the automatically generated transcriptions are compared with manual transcriptions.

Numbers in the first matrix, the “Replacement Confusion Matrix,” (B.1) state the number of times the phone along the horizontal axis (automatic transcriptions) was matched with the phone along the vertical axis (manual transcriptions). All non-zero values off of the diagonal represent disagreements between the automatic and manual transcriptions.

The “Context Confusion Matrix,” (B.2) is generated for a particular phone from the automatically-generated transcription. The matrix can also be constrained so that it only displays occurrences where the specified automatically-generated phone was aligned with a particular phone in the manually-generated transcription. B.2 shows the “eh-*” context confusion matrix for the situations where the “eh” phone of the automatically-generated transcription was aligned with “*” (any phone) of the manually-generated transcription. In contrast, a context confusion matrix could be generated for situations where the “eh” phone of the automatically-generated transcription was aligned with “aa” from the manual transcription. This “eh-aa” context confusion matrix could be used to determine in which contexts “eh” should be expanded to “(eh | aa)” in a pronunciation network.

The vertical axis in B.2 represents the phone that preceded a particular instance of “eh” in the automatic transcriptions, and the horizontal axis represents the phone that followed the same instance of “eh.” The value at a given coordinate represents the number of times a particular context occurred. For example, the sequence “w eh l” occurred 12 times, and the sequence “dh eh m” occurred 8 times.

If B.2 were the “eh-aa” matrix, the previously mentioned 12 would mean that the automatically-generated sequence “w eh l” was aligned with the manually-generated sequence “w aa l” 12 times.

Phones from Automatic Transcriptions

[illegible]

B.2 Context Confusion Matrix for the Phone “eh”

Context Confusion Matrix

Phone Following “eh”

[illegible]

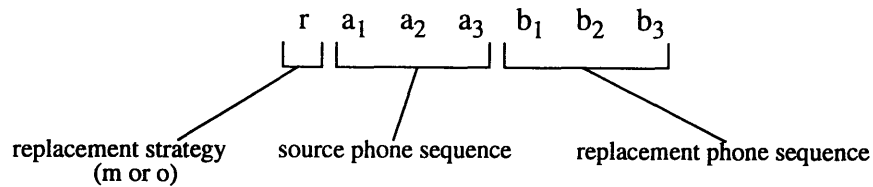
Phones Preceding “eh”

Appendix C

Phonological Rules

C.1 Rule Format

Rule Structure Definitions:



Rules in C.2 are formatted as in the figure above. The replacement strategy, *r*, can be either “o” for a rule that adds an option to a network, or “m” for a rule that makes a mandatory change to a network by eliminating or replacing an option. The *a*’s specify the source sequence to match, while the *b*’s specify new sequence.

wildcard symbols:

*	match any phone
#vc	match voiced consonant (‘a’ positions only)
#uc	match unvoiced consonant (‘a’ positions only))
>	phoneme to the right (used at position a ₁ only)
<	phoneme to the left (used at position a ₃ only)

special symbols:

.	phoneme from a _i position (used at position b _i)
[]	make phone optional (b positions only)
/	delete phone (b positions only)
+	insert a phone (b positions only)

Example 1: Standard Option Adding

original network: (x0 y0 z0)
rule applied: o x0 y0 z0 x1 y1 z1
yields network: ((x0 y0 z0) | (x1 y1 z1))

Example 2: Standard Option Replacement

original network: (x0 y0 z0)
rule applied: m x0 y0 z0 x1 y1 z1
yields network: (x1 y1 z1)

Example 3: Option Elimination

original network: (x y z)
rule applied: m x y z x / z
yields network: (x z)

Example 4: Use of Wildcards

original network: (x0 y0 z0)
rule applied: m x0 y0 * x1 y1 *
yields network: (x1 y1 z0)

Example 5a: Use of Wildcard, producing a side-effect (can be used as a feature to decrease network complexity, but when used carelessly produces unwanted options)

original network: (x0 y0 (z0 | w0))
rule applied: m x0 y0 z0 x1 y1 *
yields network: (x0 y0 (z0 | w0))
(x1 y1 w0) is allowed as a side-effect

Example 5b: Use of Wildcard, avoiding side-effects

original network: (x0 y0 (z0 | w0))
rule applied: m x0 y0 z0 x1 y1 z0 OR m x0 y0 z0 x0 y0 .
yields network: ((x0 y0 w0) | (x1 y1 z0))
(x1 y1 w0) is allowed as a side-effect

Example 6: < and > Wildcards (useful when “a₂” is some wildcard)

original network: (x0 y0 y0), y0 is a voiced consonant
rule applied: m x0 #vc < . y1 .
yields network: (x0 y1 y0)

Example 7: Optionalizing with []

original network: (x0 y0 z0)
rule applied: m x0 y0 * . [] *
yields network: (x0 [y0] z0)

Example 8: Using + for inserting a phone (phone need not be optional as shown)

original network: (x0 y0 z0)
rule applied: m x0 y0 * . +[w1] *
yields network: (x0 [w1] y0 z0)

C.2 Manually-chosen Rules

```

o d y uw / jh uw
o * t dh * [] dh
o * t d * [] d
o * z s * [] s
o * z sh * [] sh
o * s sh * [] sh
m * #vc < * [] .
m * #uc < * [] .
o * ah * * ax *
o * ao * * ow *
o * aa * * ao *
o * eh * * ey *
o * ih * * iy *
o * er * * r *
o * ax * * eh *
o ey d ih . dx .
o ey d eh . dx .
o ih d ey . dx .
o eh d ey . dx .
o r d er . dx .
o ao t ih . dx .
o ax t ih . dx .
o eh t ih . dx .
o r t ah . dx .

```

```

o w ah l . ih .
o w ih n . eh .
o ax ax ax . d .
o dx dx * . d *
o * dh dh * cl .
o y ow r y er /
o * r r * ow .
o ax v dh . [] .
o ax v dh . th /
o n vcl vcl . d .
m * t t * [] t
o f t er . d .
o k t ax . d .
o t t ax . d .
o ao t er . dx .
m * t n * [] .
m s t s . [] .
m * #cl * * +[cl] *
m * #vcl * * +[vcl] *
m * dx * * +[cl] *
m * dx * * +[vcl] *
m * d * * [] *
m * dx * * [] *

```


References

- [1] Cambridge University Engineering Department Speech Group and Entropic Research Laboratories, Inc. "HTK: Hidden Markov Model Toolkit V1.5 User and Reference Manual." December 7, 1993.
- [2] L.R. Rabiner and B. H. Juang. "An Introduction to Hidden Markov Models." *IEEE ASSP Magazine*. January, 1986
- [3] Bob Weide et al. "Carnegie Mellon Pronouncing Dictionary" (cmudict.0.4). unrestricted distribution by Carnegie Mellon University. 1995
- [4] Paul S. Cohen and Robert L. Mercer. "The Phonological Component of an Automatic Speech-Recognition System." *Speech Recognition*. D. Raj Reddy, ed., Academic Press, Inc., 1974
- [5] Egidio P. Giachin, Aaron E. Rosenberg, and Chin-Hui Lee. "Word Juncture Modeling Using Phonological Rules for HMM-based Continuous Speech Recognition." *Computer Speech and Language*. 5, 155-168, 1991
- [6] Beatrice T. Oshika, Victor W. Zue, Rollin V. Weeks, Helene Neu, and Joseph Aurbach. "The Role of Phonological Rules in Speech Understanding Research." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-23, No. 1, February, 1975
- [7] Charles Hoequist Jr. and Francis Nolan. "On an Application of Phonological Knowledge in Automatic Speech Recognition." *Computer Speech and Language*. 5, 133-153, 1991